symmetry

MDPI

Article

SAAE-DNN: Deep Learning Method on Intrusion Detection

Chaofei Tang [1], Nurbol Luktarhan [2,*] and Yuxin Zhao [1]

[1] College of Software, Xinjiang University, Urumqi 830000, China; fei1318281@stu.xju.edu.cn (C.T.); zhaoyuxin@stu.xju.edu.cn (Y.Z.)
[2] College of Information Science and Engineering, Xinjiang University, Urumqi 830000, China
* Correspondence: nurbol@xju.edu.cn

Received: 16 September 2020; Accepted: 12 October 2020; Published: 15 October 2020

Abstract: Intrusion detection system (IDS) plays a significant role in preventing network attacks and plays a vital role in the field of national security. At present, the existing intrusion detection methods are generally based on traditional machine learning models, such as random forest and decision tree, but they rely heavily on artificial feature extraction and have relatively low accuracy. To solve the problems of feature extraction and low detection accuracy in intrusion detection, an intrusion detection model SAAE-DNN, based on stacked autoencoder (SAE), attention mechanism and deep neural network (DNN), is proposed. The SAE represents data with a latent layer, and the attention mechanism enables the network to obtain the key features of intrusion detection. The trained SAAE encoder can not only automatically extract features, but also initialize the weights of DNN potential layers to improve the detection accuracy of DNN. We evaluate the performance of SAAE-DNN in binary-classification and multi-classification on an NSL-KDD dataset. The SAAE-DNN model can detect normally and attack symmetrically, with an accuracy of 87.74% and 82.14% (binary-classification and multi-classification), which is higher than that of machine learning methods such as random forest and decision tree. The experimental results show that the model has a better performance than other comparison methods.

Keywords: intrusion detection; attention mechanism; stacked autoencoder; DNN; classification; NSL-KDD; deep learning

1. Introduction

With the development and perfection of Internet technology, the Internet is playing an increasingly significant role in our work and life. However, in the process of using and interacting with the Internet, a large amount of data are generated, processed, and exchanged. These data have become the targets of illegal activities, which has posed a major threat to network security [1]. Luckily, the intrusion detection system (IDS) [2] can solve these problems well. As an active security technology, IDS monitors networks or hosts and alerts when attacks are detected. Network security can be better ensured via intrusion detection methods in which the network attack behavior can be learned through data analysis and modeling. How to identify all kinds of network malicious traffic, especially unknown network malicious traffic, is an unavoidable key problem. Network traffic can be divided into normal traffic and malicious traffic. Besides this, network traffic can be divided into five categories: normal, Probing attacks (Probe), Denial of Service attacks (DoS), User to Root attacks (U2R), and Root to Local attacks (R2L). Therefore, intrusion detection can be regarded as a classification problem [3]. By improving the performance of the classifier in effectively identifying malicious traffic, the accuracy of intrusion detection can be greatly improved. Improving the accuracy of IDS is realized by improving the performance of the classifier in effectively identifying malicious traffic.

In recent years, more and more intrusion detection methods have been applied to intrusion detection, including machine learning methods, ensemble learning methods, and deep learning methods. Machine learning methods first select features, and then use classifiers to detect intrusions, such as random forest (RF) [4], decision tree (DT) [5], and support vector machine (SVM) [6]. Deep learning methods can automatically extract features and classify to realize intrusion detection, such as autoencoders [7], long short term memory (LSTM) [8], and deep neural networks (DNN) [9]. The ensemble learning method uses various ensemble and hybrid technologies for intrusion detection, including bagging [10], boosting [11], stacking [12], and combined classifier methods [13].

Machine learning methods are widely used in intrusion detection [14–17]. However, machine learning methods often emphasize feature selection. When facing high-dimensional data, there are difficulties in feature selection, which cannot effectively solve the classification problem of massive intrusion data, resulting in low recognition accuracy and other problems. Deep learning has made great progress in computer vision, natural language processing, and other fields, and has gradually been applied to other fields of artificial intelligence. In recent years, many deep learning methods have been applied to intrusion detection. Deep learning methods can automatically extract high-level latent features without manual intervention [18]. Unlike principal component analysis (PCA) and chi-square feature selection, which rely heavily on artificial feature extraction and mainly rely on experience and luck, the results are not ideal. Due to the characteristics of the large quantity, high dimension, and complex structure of network traffic, the traditional ML technology has limited computational complexity, and there are still deficiencies in learning complex nonlinear relationships in large datasets.

Considering the above factors, to improve the accuracy of intrusion detection in binary-classification and multi-classification, we propose a new network intrusion detection model SAAE-DNN, which mainly combines stacked autoencoder(SAE) [19], attention mechanism [20], and DNN [21], as shown in Figure 1. SAE is used to compress the input to the latent layer, but the features will still have redundancy, and then the attention mechanism is introduced to learn the latent layer features. To date, we have completed the key features extraction of network traffic through SAAE encoder. Finally, the latent layer of the SAAE encoder is connected with DNN and classified by softmax function to realize intrusion detection. To verify the effectiveness of the SAAE-DNN model, a comprehensive evaluation is carried out on the NSL-KDD dataset, and the best results are obtained, with the prediction accuracy reaching 87.74% and 82.14% (binary-classification and multi-classification).

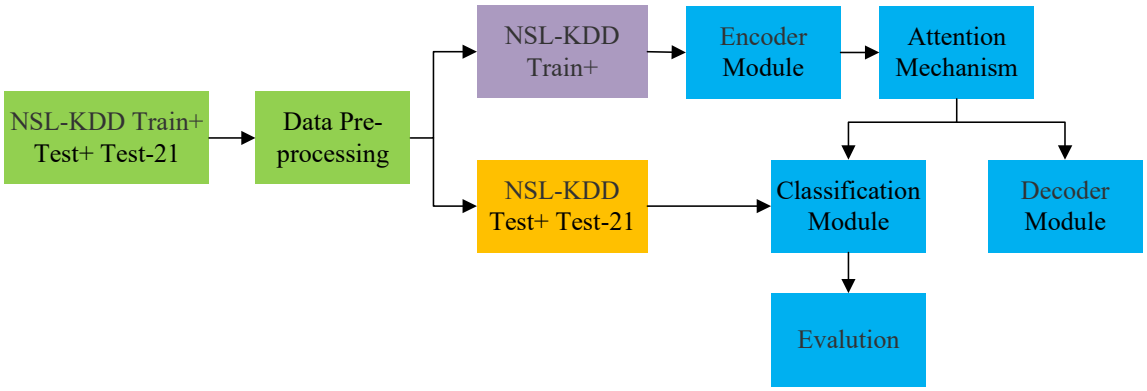


Figure 1. The framework of the proposed intrusion detection method.

The main contributions and findings of this paper are as follows:

- A deep learning model SAAE-DNN is proposed, which consists of SAE, attention mechanism, and DNN. SAAE-DNN improves the accuracy of IDS and provides a new research method for intrusion detection;
- We introduce attention mechanism to highlight the key inputs in the SAE model. The attention mechanism learns the latent layer features of SAE, and the obtained feature information is reasonable and accurate;

- We use a real NSL-KDD dataset to evaluate our proposed network. The experimental results show that SAAE-DNN has better performance than traditional methods.

2. Related Work

Deep learning has achieved success in image recognition, speech detection, and other fields, and has become the preferred solution to many problems. In recent years, this method has been gradually applied to the field of intrusion detection and achieved remarkable detection results. Javaid, A. Y. et al. [22] propose a deep learning based approach for developing such an efficient and flexible NIDS. They use Self-taught Learning (STL) on NSL-KDD dataset for network intrusion. In the case of two classifications and multiple classifications, the accuracy rate is 88.39% and 79.10%, respectively. Yin, C. et al. [23] propose an intrusion detection method using recurrent neural network (RNN-IDS). The accuracy rate is 81.29% on NSL-KDD dataset, which is higher than the machine learning methods proposed by predecessors such as random forest and support vector machine.

Feature selection technology reduces the dimension of feature space and identifies important features by removing irrelevant and redundant attributes, which is very important in intrusion detection. Tom et al. [24] use filter and wrapper feature selection methods for feature selection. The framework is evaluated on the UNSW-NB15 dataset. The results show that one of them uses a filtering sorting method to carry out feature features with the highest accuracy of 88%. Mighan, S. N. et al. [25] combines the advantages of deep network and machine learning methods, using an SAE network for latent feature extraction, followed by several machine learning methods for intrusion detection, such as SVM, random forest, decision tree, and Naive Bayes. Jo, W. et al. [26] propose three preprocessing methods to make CNN kernel fully reflect the characteristics of the network. The proposed preprocessing method is based on the philosophy of field-to-pixel, each field converted into an image takes up one pixel, and then the pixels are converted into convolution layers through pooling layers. The method reflects the advantages of CNN by extracting the convolution features of each pixel. One of the most promising research trends is to incorporate an attention mechanism into the framework of depth learning [27]. The attention mechanism is very effective in image classification [28] and natural language processing [29], which draws lessons from the human visual attention mechanism. Human vision obtains the target area to be focused by scanning the global image quickly, which is the focus of attention. Then, more attention resources are put into focus to obtain more detailed information about the target and inhibit useless information. Yang, T. et al. [30] use bidirectional recurrent neural networks with attention mechanism to extract potential features and give interpretable results in identifying dominant attributes.

3. Background

3.1. AutoEncoder

An autoencoder (AE) is a kind of neural network that uses encoding and decoding process for unsupervised learning, which is mostly used for feature extraction and dimension reduction. AE is a structure that contains symmetry. The AE structure includes an input layer, a latent layer, and an output layer. As shown in Figure 2, the input layer and the output layer are equal in size, and the size of the latent layer must be smaller than that of the input layer [31].

The input vector is x, the latent vector is $e \in [0,1]^d$, and the constructed vector is $\tilde{x} \in [0,1]^D$. The encoding process from input layer to latent layer is

$$e = f_\theta(x) = s(Wx + b). \tag{1}$$

and the decoding process from latent layer to output layer is

$$\tilde{x} = g_{\theta'}(e) = s(W'e + b'), \tag{2}$$

where W and W' represent the input-to-latent and the latent-to-output weights, respectively, b and b' are the bias vectors of the input layer and the latent layer, respectively, and f_θ and $g_{\theta'}$ are the activation functions of the latent layer neurons and the output layer neurons. Parameters W, W', b and b' in the AE are learned by minimizing the reconstruction error.

$$J(W, b; x, \tilde{x}) = \frac{1}{2}\left\|h_{W,b}(x) - \tilde{x}\right\|^2. \tag{3}$$

The above is a measurement of the error between the reconstructed $\tilde{x}$ and the input x for a single sample. In a training set containing D samples, the cost function of the AE is defined as

$$\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(W_{ji}^{(l)}\right)^2 = \left[\frac{1}{D}\sum_{i=1}^{D}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - \tilde{x}^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(W_{ji}^{(l)}\right)^2. \tag{4}$$

where D represents the total number of samples, s represents the number of nodes in layer l, λ is a weight attenuation parameter, and the reconstruction error of each training sample is the square error. The second item is introduced to reduce the size of the weight, which helps to prevent overfitting.

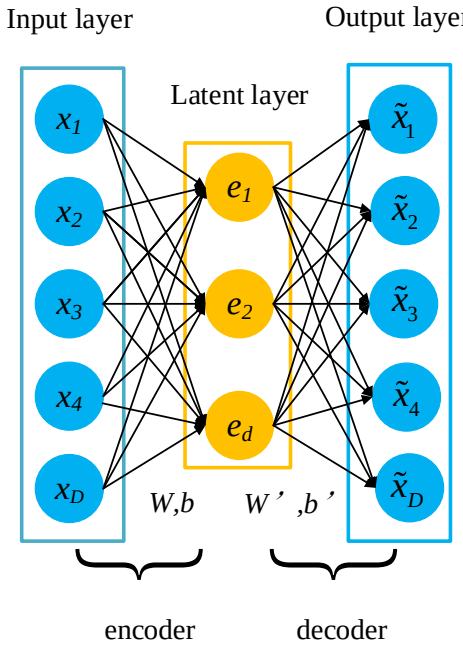


Figure 2. The structure of AE.

3.2. *Stacked AutoEncoder*

SAE is also a structure that contains symmetry. SAE is generated by AEs, stacked layer by layer. Once the single-layer AE is trained, as shown in Figure 3, the second AE is trained using the latent layer from the first AE. By repeating this process, we can create an SAE of any depth [32].

AEs are stacked to achieve greedy hierarchical learning, where the lth latent layer is used as input to the l + 1th latent layer in the stack. The results generated by the SAE are used to pre-train the weights of fully connected DNN instead of randomly initializing the weights. This method is helpful for the model to initialize parameters close to good local minimum values and improve the optimization effect. This shows that the convergence of the method is smoother and the overall performance is higher in classification tasks.

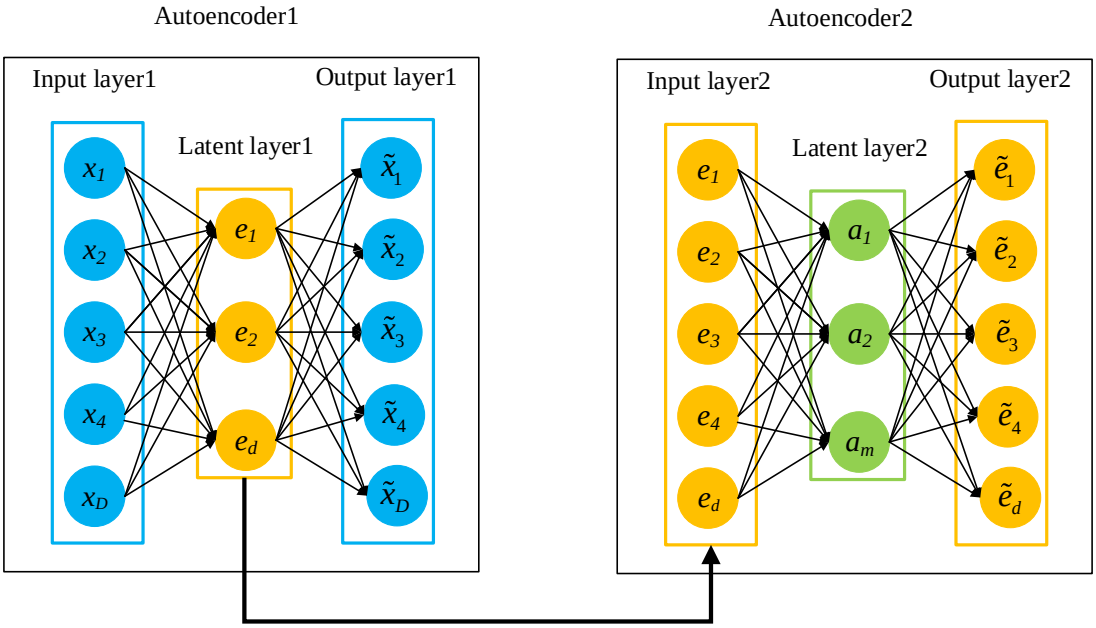


Figure 3. Example of SAE formed by two single AEs.

3.3. Attention Mechanism

The AE reduces the dimension of the data layer by layer and represents the data in a compressed form, so we mainly focus on the latent layers of the AE. By reconstructing the data, it learns to find useful features in the data that are difficult to find. Although the data are compressed by AE, the data representation in AE may still contain redundant features. Features are important for prediction, but their importance in the whole dataset is not equal. The attention mechanism layer contains a value vector, and each value in the vector contains the importance value of the corresponding feature. Through weighted summation, a small number of important features are selected from a large amount of feature information focusing on these features. The formulas are as follows

$$M = tanh(W_a x' + b_a), \tag{5}$$

$$\alpha_i = softmax(M_i), \tag{6}$$

$$v = \sum_{i=0}^{D} x' \alpha_i^T. \tag{7}$$

where x' refers to the output of the encoder in AE, W_a is the weight, b_a is the offset value, and α_i is the probability distribution of M_i normalized by softmax. Finally, the probability distribution is taken as the weight and summarized with x' to obtain a more representative feature vector v, which focuses on important features. The attention mechanism layer eliminates unnecessary features and gradually improves the performance of the network [33].

3.4. Deep Neural Network

The essence of deep learning is to learn more useful features by constructing multiple hidden layers, thus obtaining higher accuracy [34]. The basic structure of DNN is composed of an input layer, an output layer, and one or more hidden layers. Each layer of DNN consists of one or more artificial neurons, so that these neurons are completely connected from one layer to another. In addition, information is processed through DNN feedforward mode, i.e., from the input layer to output layer through the hidden layer. Figures 4 and 5 show the architectures of DNN for binary-classification and multi-classification.

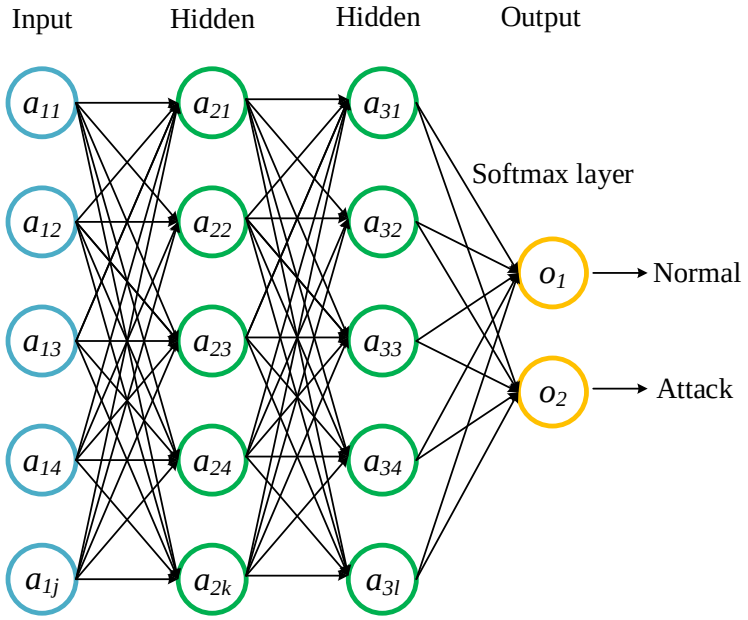


Figure 4. DNN ((binary-classification).

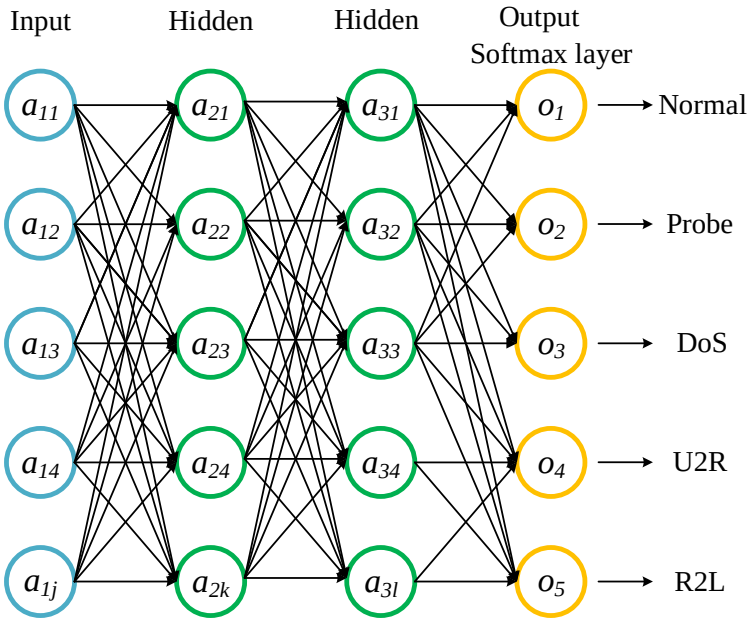


Figure 5. DNN (multiclass-classification).

4. The Proposed Intrusion Detection Method

The framework of the proposed SAAE-DNN is shown in Figure 6. SAAE-DNN mainly includes three stages: (1) Data preprocessing: scaling data by using min-max normalization technology. Then, the one-hot encoding technique is used to convert the categorical features to the values 0 and 1. Finally, the features with zeros exceeding 80% are removed. (2) Attention mechanism is added to SAE, and the SAAE model is trained to realize feature extraction. (3) Detecting attacks: using the latent layer of the SAAE to initialize the weight of DNN, the trained DNN classifier is used to detect attacks.

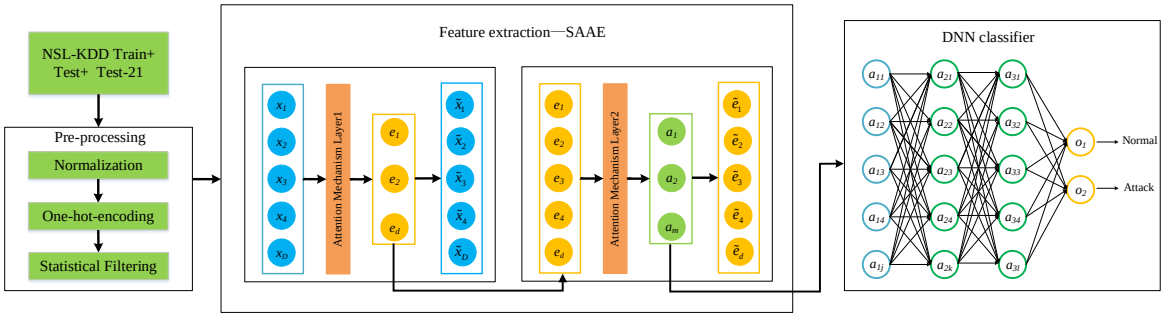


Figure 6. The flow chart of the proposed intrusion detection method.

4.1. Data Preprocessing

Data preprocessing is necessary for IDS. It includes three units: data normalization, one-hot-encoding, and statistical filtering.

4.1.1. Data Normalization

When using deep learning, an important issue is that some dimensions have completely different scales. The NSL-KDD dataset consists of 41 dimensions, whose values vary greatly. In this paper, the min–max normalization method [35] is adopted to reduce the different scales of dimensions. Through the linear transformation of the original data, it is scaled to the interval [0, 1]. The min–max values perform the data conversion using the following equation

$$\tilde{z}_{fj} = \frac{z_{fj} - min(z_f)}{max(z_f) - min(z_f)}. \tag{8}$$

where $min(z_f)$ and $max(z_f)$, respectively, represent the minimum and maximum values of the fth numeric feature z_f, and $\tilde{z}_{fj}$ indicates that the normalized feature value is between [0,1].

4.1.2. One-Hot-Encoding

As a simple and effective encoding technology, one-hot-encoding [36] is the most commonly used method to deal with the numeralization of categorical features. It can convert the value of each categorical feature into a binary vector, in which there is only one element with a value of 1 and all other elements are zero. An element with a value of 1 indicates the existence of possible values corresponding to the categorical feature. There are a total of three categorical features in NSL-KDD: *protocol_ type*, *service*, *flag* (*z2*, *z3*, *z4*, respectively). For example, the feature *protocol_ type* has a total of three attributes: *tcp*, *udp* and *icmp*. Using one-hot-encoding, *tcp* can be encoded as (1, 0, 0), *udp* can be converted to (0, 1, 0), and *icmp* can be converted to (0, 0, 1). Similarly, *service* and *flag* are also converted into one-hot-encoding vectors. Overall, three categorical features *protocol_ type*, *service*, *flag* are mapped into 84-dimensional binary values.

4.1.3. Statistical Filtering

This unit is used to filter out irrelevant features. For each continuous feature, the percentage of zeros is calculated. Figure 7 illustrates the distribution of zeros of each numerical feature in the KDDTrain+ dataset. According to [7] , features with zeros greater than 80% will be excluded. Specifically, 20 numerical features will be discarded (the feature where the bar shape is higher than the red line in Figure 7). The other 18 numerical features are combined with 84 one-hot-encoding vectors, and a total of 102-dimensional feature vectors are used as inputs for the next stage.

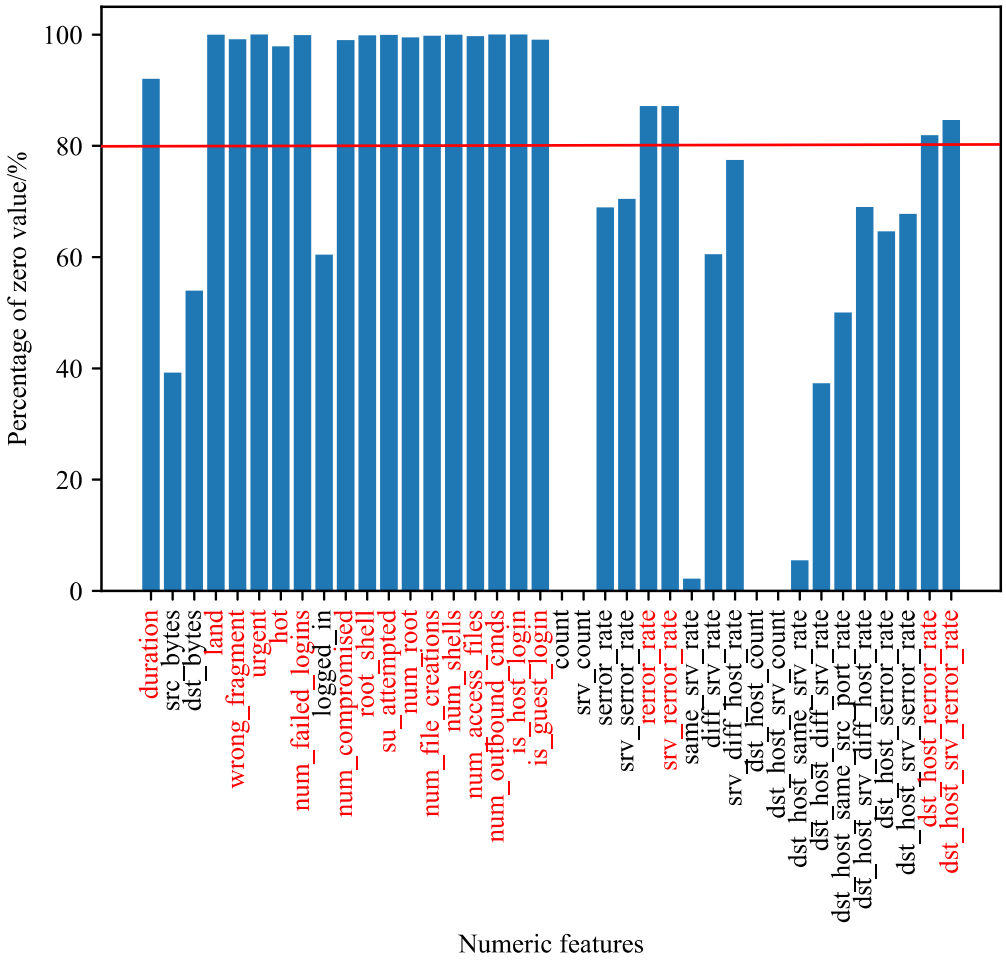


Figure 7. Bar chart contained in 38 numeric variables of KDD Train+. The features where the bars are higher than the red line indicate that the zero values are greater than 80% and are removed. The removed feature name has been marked red.

4.2. Proposed Attention Autoencoder (AAE) and Stacked Attention AutoEncoder (SAAE)

The structure of the attention-autoencoder (AAE) is shown in Figure 8. We add an attention mechanism layer between the encoder and the latent layer. The original input data passes through the encoder, and the data have been compressed. The layer calculates the attention vector of each simplified feature. The attention vector and the feature are multiplied to generate data input to the latent layer. When the attention vector finds that a specific feature does not contribute to the prediction, it sets the specific value in the vector to 0, thus causing the network to forget the feature. The classification module obtains the data representation of the latent layer and outputs the final predicted value. Similarly, the stacked-attention-autoencoder (SAAE) model we proposed has two AAEs stacked, and the structure is shown in Figure 9.

In the experiment, we use an SAAE composed of two stacked AAEs. There are 90 and 80 neurons in the hidden layer of each AE, respectively. The SAAE is configured to reduce the dimension of the original input data layer by layer, thus reducing noise while retaining important information.

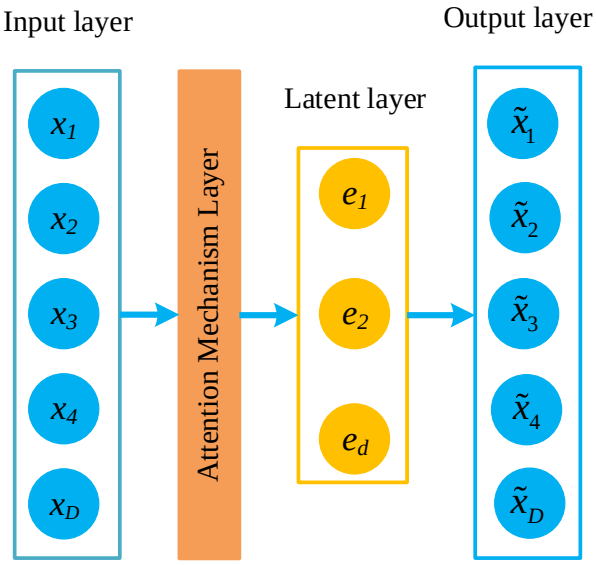


Figure 8. The structure of AAE.

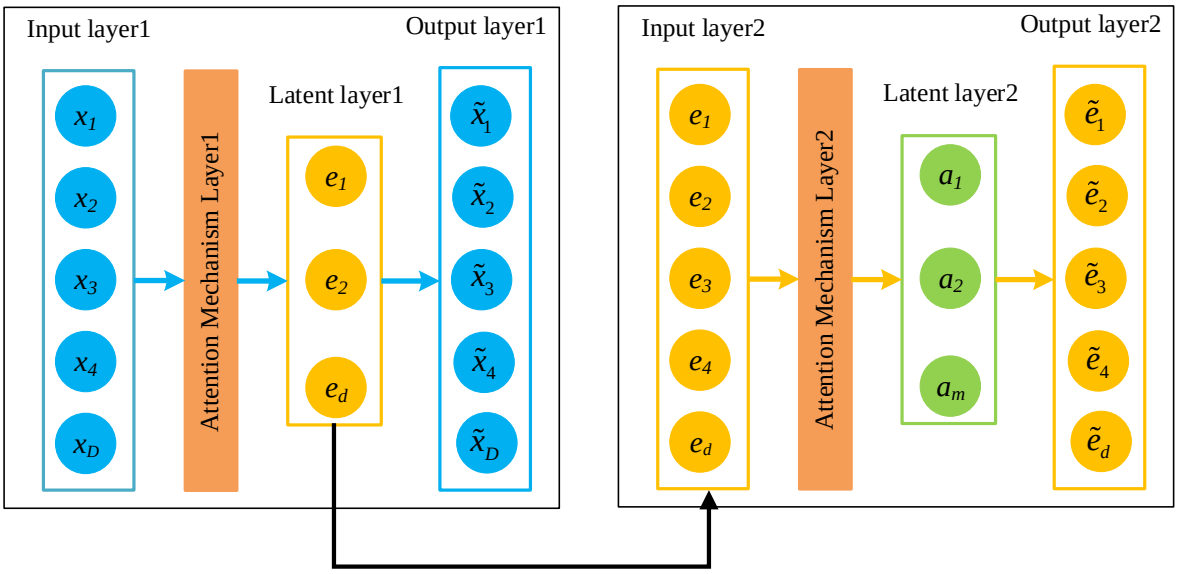


Figure 9. Example of SAAE formed by two single AAEs.

4.3. Classification

A deep learning method based on SAAE and DNN is developed to detect the normal and attack categories of the NSL-KDD dataset (Normal, Probe, DoS, U2R, and R2L). DNN is applied by us to detect attacks. The SAAE encoder can automatically extract features, so the weights of the trained SAAE encoder are used to initialize the weights of DNN hidden layer, and the activation function of output layer in DNN is softmax. Finally, the test samples are input into the trained SAAE-DNN classifier to detect attacks. See the following Algorithm 1 for details.

Algorithm 1 The Intrusion Detection Algorithm with SAAE and DNN

Input: Training dataset KDDTrain+, Testing dataset KDDTest+ and KDDTest-21;
Output: Classification results: accuracy, precision, recall, and F1-score;

Step1: Data preprocessing
    $z_f' = normalization(z_f)$;
    $z_2', z_3', z_4' = one - hot(z_2, z_3, z_4)$;
    Statistical filtering;
EndStep
Step2: Feature extraction
    The network structure dimension of the SAAE is 102-90-80-90-102;
    Train the SAAE using the NSL-KDD Train+ dataset and minimize the reconstruction error according to Equation (4);
EndStep
Step3: Classification
    The weights of the latent layer of trained SAAE is used to initialize the weight of the DNN;
    Train the DNN classifier;
    Testing dataset KDDTest+ and KDDTest-21 are input into the trained SAAE-DNN classifier to detect attacks;
EndStep
Return the classification result.

5. Experimental Results and Analysis

5.1. Experimental Environment

We conduct experiments to evaluate the performance of the proposed SAAE-DNN. The proposed SAAE-DNN was implemented in the Keras environment with 64 GB RAM, RTX-2080Ti GPU and 64-bit Ubuntu18.04 operating system.

5.2. Performance Metrics

To effectively evaluate the performance of the proposed intrusion detection method, four performance measures, such as accuracy, precision, recall and F1-score, are adopted. As shown in Table 1, these performance measures are calculated according to the confusion matrix of network attack classification [37]. True Positive (TP) is the number of records where attack traffic is correctly classified as attack traffic; True Negative (TN) is the number of records in which normal traffic is correctly classified as normal traffic; false positive (FP) is the number of records that mistakenly classify normal traffic as attack traffic; False Negative (FN) is the number of records that mistakenly classify attack traffic as normal traffic.

The calculation formula is as follows

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}. \tag{9}$$

$$Precision = \frac{TP}{TP + FP}. \tag{10}$$

$$Recall = \frac{TP}{TP + FN}. \tag{11}$$

$$F1\text{-}score = 2 \times \frac{precision \times recall}{precision + recall}. \tag{12}$$

Table 1. confusing matrix.

	Predicted Attack	**Predicted Normal**
Actual attack	TP	FN
Actual normal	FP	TN

5.3. Description of the Benchmark Dataset

At present, only a few benchmark datasets can evaluate the effectiveness of the IDS. The NSL-KDD dataset [38] overcomes many inherent problems existing in KDD '99 dataset and is an improvement on the KDD' 99 dataset. The NSL-KDD dataset is suitable for evaluating various attacks [39], so we chose it to evaluate the detection performance of the proposed model. The NSL-KDD dataset eliminates redundancy and duplicate records in the KDD '99 dataset, and the number of samples in training and the testing dataset is more reasonable. The NSD-KDD dataset contains normal and attack records. Attacks include four types of record: Probe, DoS, U2R, and R2L. We use KDDTrain+ as the training set and KDDTest+ and KDDTest-21 (deleting 21 records that are easy to classify by common classifiers in KDDTest+ dataset) as the testing sets. The number of records for each type in the NSL-KDD dataset is shown in Table 2.

As can be seen from Table 2, NSL-KDD training sets are imbalanced. Normal accounts for 53.46%, 43.07% and 18.16% in KDDTrain+, KDDTest+ and KDDTest-21 m respectively, which is much higher than that of U2R, which accounts for 0.04%, 0.89% and 1.69%m respectively.The NSL-KDD dataset contains a total of 39 attacks. A new set of attacks was introduced into the testing dataset. These new attacks do not appear in the training dataset and are shown in bold. In addition, KDDTest-21 removes some attacks that are easily detected correctly. All of these add to the difficulty of testing on the NSL-KDD test set.

Table 2. Class distribution of the NSL-KDD dataset.

Category	**Attack**	**Training Dataset KDDTrain+**	**Testing Dataset KDDTest+**	**KDDTest-21**
Normal	normal	67,343	9711	2152
Subtotal		67,343	9711	2152
Subtotal Percentage		53.46%	43.07%	18.16%
Probe	ipsweep	3599	141	141
	satan	1493	735	727
	portsweep	2931	157	156
	nmap	3633	73	73
	saint	/	319	309
	mscan	/	996	996
Subtotal		11,656	2421	2402
Subtotal Percentage		9.25%	10.74%	20.27%

Table 2. *Cont.*

Category	Attack	Training Dataset KDDTrain+	Testing Dataset KDDTest+	KDDTest-21
DoS	neptune	41,214	4657	1579
	smurf	2646	665	627
	back	956	359	359
	teardrop	892	12	12
	pod	201	41	41
	land	18	7	7
	apache2	/	737	737
	mailbomb	/	293	293
	processtable	/	685	85
	udpstorm	/	2	2
Subtotal		45,927	7458	4342
Subtotal Percentage		36.46%	33.08%	36.64%
U2R	buffer_overflow	30	20	20
	rootkit	10	13	13
	loadmodule	9	2	2
	perl	3	2	2
	httptunnel	/	133	133
	ps	/	15	15
	sqlattack	/	2	2
	xterm	/	13	13
Subtotal		52	200	200
Subtotal Percentage		0.04%	0.89%	1.69%
R2L	guess_passwd	53	1231	1231
	warezmaster	20	944	944
	imap	11	1	1
	multihop	7	18	18
	phf	4	2	2
	ftp_write	8	3	3
	spy	2	/	/
	warezclient	890	/	/
	named	/	17	17
	sendmail	/	14	14
	xlock	/	9	9
	xsnoop	/	4	4
	worm	/	2	2
	snmpgetattack	/	178	178
	snmpguess	/	331	331
Subtotal		995	2754	2754
Subtotal Percentage		0.79%	12.22%	23.24%
Total		125,973	22,544	11,850

5.4. Experimental Step

We conduct experiments on the NSL-KDD dataset to evaluate the performance of the model and compared the results of the model with SAAE-DNN and commonly used machine learning methods. The structure of SAAE is shown in Figure 9, two AAEs are stacked together. The number of neurons in the latent layer of each AAE is 90 and 80, respectively, the optimizer of each AAE is Adam, and the activation function of each layer is ReLU. The number of neurons in each hidden layer of DNN is [50, 25, 10], the activation function of each hidden layer is also ReLU, the activation function of the output layer is softmax for classification, and the optimizer is Adam. Two important parameters need to be set, including learning rate and epoch.

When the learning rate is too high, the network will oscillate during the training process, resulting in non-convergence. If the learning rate is too small, the convergence will be slow. In Keras, the default learning rate for the Adam optimizer is 0.001, so my candidate learning rate is 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005. SAAE is used for feature extraction. We can classify these by adding a softmax after the latent layer of the SAAE, to directly detect the influence of the learning rate on SAAE. The choice of the epoch is mainly considered from the direction of training loss. For the selection of these two parameters in DNN, the learning rate is determined according to the accuracy of the training set, and the epoch is also determined according to the training loss.

The training set is divided into five equal parts, and after five-fold cross-validation, we find the best parameters. First of all, for SAAE, as can be seen from Figure 10, when the learning rate is equal to 0.05, the accuracy is the highest at 89.82%. As can be seen from Figure 11, the training loss does not decrease after the epoch is greater than 100, so the epoch of SAAE is set to 100. The selection of DNN parameters is also based on the same considerations. As can be seen from Figure 12, when the learning rate is 0.006, the highest accuracy rate is 98.12%, which is much higher than the accuracy rate of 56.80% when the learning rate is 0.01. As shown in Figure 13, the training loss decreases very quickly in the first few epochs. After the epoch reaches 100, the training loss tends to stabilize, so the epoch is set to 100.

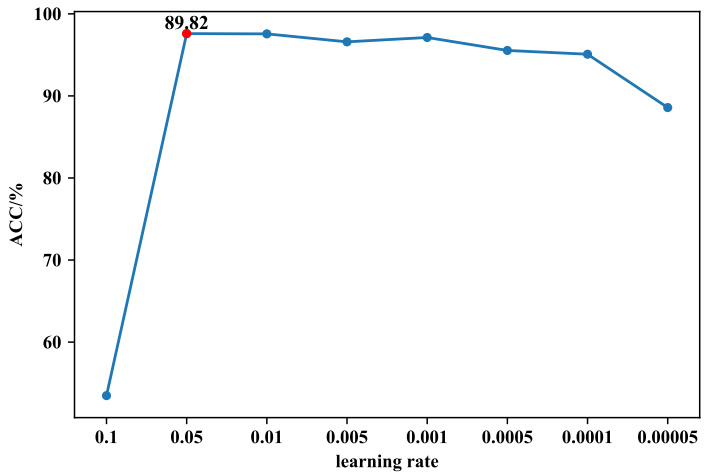


Figure 10. The accuracy of SAAE-softmax changes when different learning rates are set.

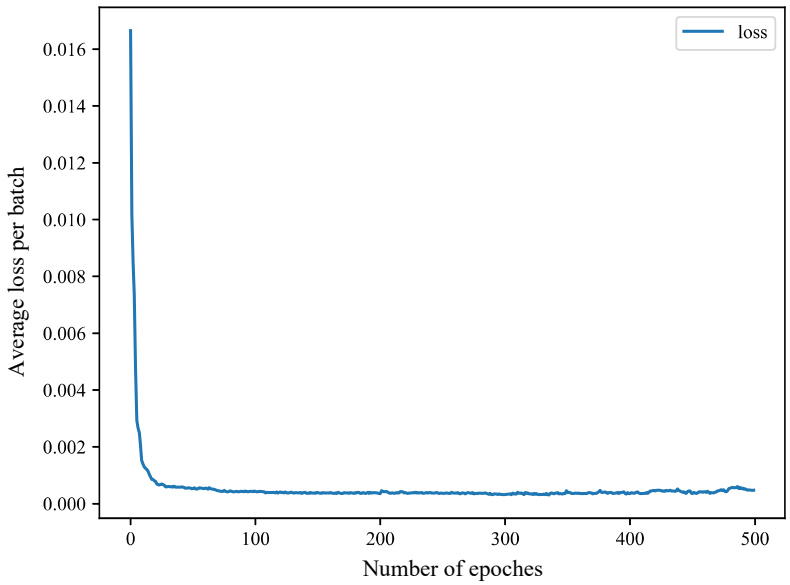


Figure 11. The training loss of the SAAE.

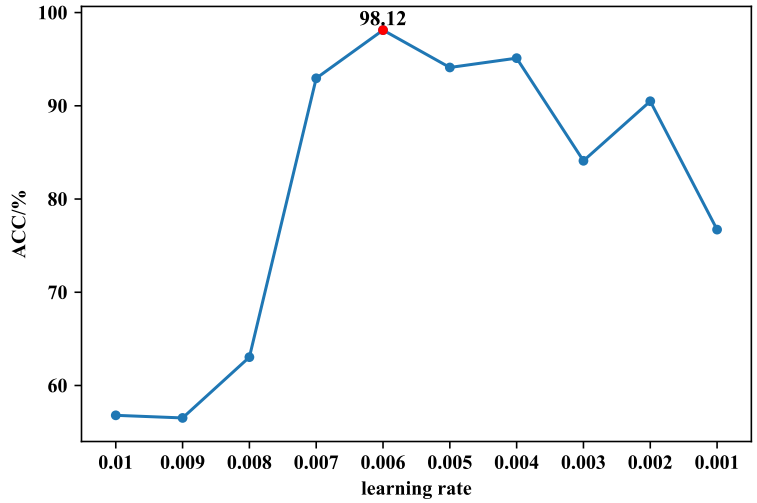


Figure 12. The accuracy of SAAE-DNN changes when different learning rates are set.

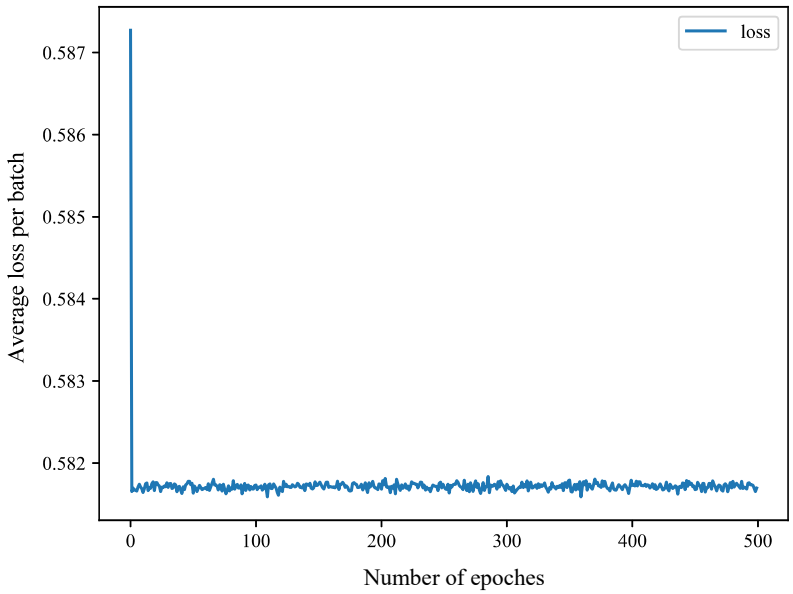


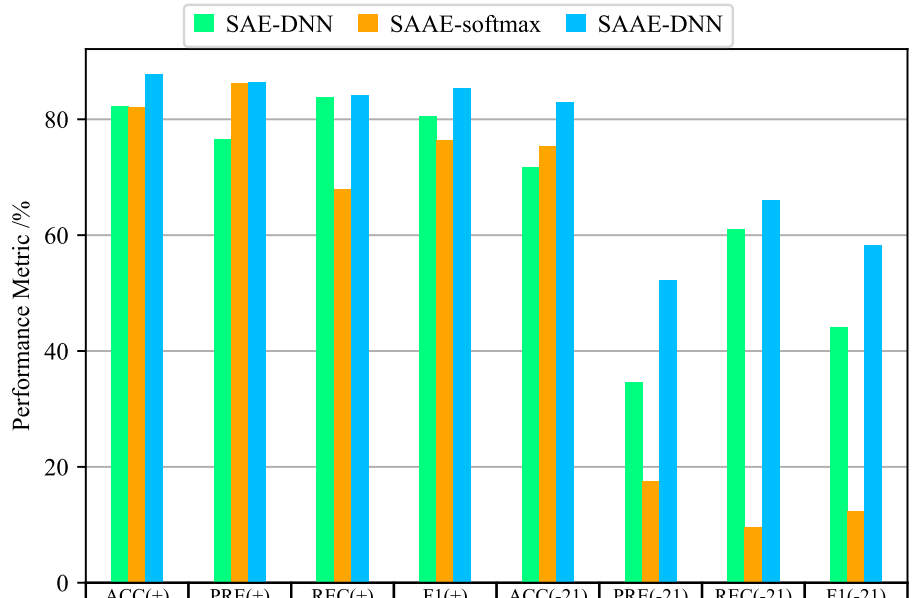
Figure 13. The training loss of the DNN.

Different numbers of AAEs are stacked together, and the performance of SAAE is also different. We set the number of AAEs to 1, 2, 3, and 4, respectively and test them. When the number of AAE is 1, there is no stacking. In the case of binary-classification, the accuracy of AAE-DNN, SAAE-DNN (2), SAAE-DNN (3) and SAAE-DNN (4) on the training set is 97.26%, 98.12%, 95.48% and 94.73%, respectively. Where the numbers in parentheses represent the number of AAEs stacked. When two AAEs are stacked, the accuracy on the training set is as high as 98.12%, so we choose to stack two AAEs.

5.5. Results and Discussion

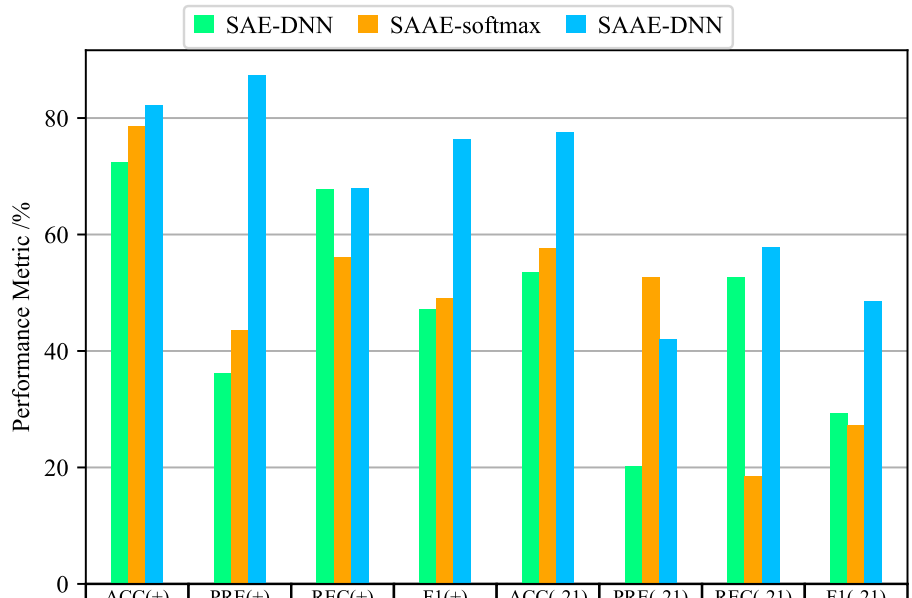
Figures 14 and 15 show the evaluation results of SAE-DNN, SAAE-softmax, and SAAE-DNN on KDDTest + and KDDTest-21 test sets, and evaluate the binary-classification and multi-classification results from four evaluation indexes of accuracy, precision, recall, and F1-score. We can see that in the binary-classifications, the accuracy of SSAE-DNN is 5.51% and 11.21% higher than that of SAE-DNN, respectively, on KDDTest + and KDDTest-21 test sets. Similarly, the precision, recall rate, and F1-score are 9.97% and 17.63%, 0.32% and 4.92%, 4.84%, and 14.15% higher, respectively. In the multi-classification results, we can also see similar results. From this group of comparisons, we can see

that SAAE can effectively extract key features and improve the intrusion detection effect after adding the attention mechanism.



	ACC(+)	PRE(+)	REC(+)	F1(+)	ACC(-21)	PRE(-21)	REC(-21)	F1(-21)
SAE-DNN	82.23	76.5	83.86	80.46	71.76	34.54	61.09	44.12
SAAE-softmax	82.14	86.28	67.89	76.38	75.38	17.44	9.6	12.39
SAAE-DNN	87.74	86.47	84.18	85.3	82.97	52.17	66.01	58.27

Figure 14. Binary-classification performance of SAE-DNN, SAAE-softmax, SAAE-DNN on NSL-KDD Test+ and NSL-KDD Test-21. (+) refers to the results on NSL-KDD Test + and (-21) refers to the results on NSL-KDD Test-21.



	ACC(+)	PRE(+)	REC(+)	F1(+)	ACC(-21)	PRE(-21)	REC(-21)	F1(-21)
SAE-DNN	72.35	36.23	67.78	47.2	53.47	20.24	52.55	29.22
SAAE-softmax	78.58	43.56	55.99	48.98	57.54	52.71	18.4	27.27
SAAE-DNN	82.14	87.28	67.89	76.37	77.57	41.93	57.72	48.56

Figure 15. Multi-classification performance of SAE-DNN, SAAE-softmax, SAAE-DNN on NSL-KDD Test+ and NSL-KDD Test-21. (+) refers to the results on NSL-KDD Test + and (-21) refers to the results on NSL-KDD Test-21.

Next, the comparison results of SSAE-DNN and SSAE-softmax are analyzed. SSAE-softmax is classified by connecting only one output layer after SAAE, where the activation function is softmax. On the KDDTest + test set, the SSAE-softmax results are good, with 5.6% and 3.56% lower accuracy than SAAE-DNN. However, on the KDDTest-21 test set, the detection effect of SSAE-softmax becomes unstable. In binary-classification, although the accuracy reaches 75.38%, precision, recall, and F1-score are only 17.44%, 9.6%, and 12.39%. From this group of comparisons, we can see that after connecting DNN behind SAAE, the detection effect is significantly improved.

To prove the superiority of SAAE-DNN in feature extraction, two classification models are constructed based on PCA-DNN and CHI2-DNN. PCA and chi-square feature selection are used for feature extraction. Table 3 shows the results of the comparison. As far as specific feature extraction algorithms are concerned, PCA and CHI2 are both feature extraction algorithms that are frequently used. In the binary-classification, the accuracy of SAAE-DNN proposed by us is 8.61% and 9.36% higher than PCA-DNN and CHI2-DNN, respectively. At the same time, in multi-classification, SAAE-DNN performs relatively better, with 10.91% and 14.25% higher accuracy than PCA-DNN and CHI2-DNN, respectively.

Table 3. Accuracy of PCA-DNN and CHI2-DNN on KDDTest+ and KDDTest-21 (ACC/%).

Methods	Binary-Classification		Multi-Classification	
	KDDTest+	KDDTest-21	KDDTest+	KDDTest-21
PCA-DNN	79.13	67.89	71.23	56.92
CHI2-DNN	78.38	72.78	67.89	59.323
SAAE-DNN	87.74	82.97	82.14	77.57

We compare SAAE-DNN with six commonly used machine learning algorithms, including Decision Tree, XGBoost, LightGBM, GBDT, Logistic Regression, and Random Forest, and the results are shown in Table 4. Specifically, we mainly analyze the accuracy of binary-classification and multi-classification. Among the six machine learning algorithms, the Decision Tree performs best, obtaining 78.34%, 62.09%, 76.50%, and 56.84% on KDDtest + and KDDTest-21, respectively, in binary-classification and multi-classification. Our proposed SAAE-DNN is 9.40%, 20.88%, 5.64%, and 20.73% higher than Decision Tree, respectively, which is higher than all the above-mentioned machine learning algorithms, especially on the KDDTest-21 test set.

Table 4. Accuracy of six machine learning algorithms on KDDTest+ and KDDTest-21 (ACC/%).

Methods	Binary-Classification		Multi-Classification	
	KDDTest+	KDDTest-21	KDDTest+	KDDTest-21
DT	78.34	62.09	76.5	56.84
XGBoost	75.85	54.059	75.62	53.68
LightGBM	77.4	57.15	68.54	44.4
GBDT	78.29	58.84	70.96	52.25
LR	75.4	53.42	75.04	52.65
RF	77.75	57.75	76.22	55.22
SAAE-DNN	87.74	82.97	82.14	77.57

5.6. Additional Comparison

To better demonstrate the performance of SAAE-DNN in intrusion detection, we compare it with six proposed intrusion detection methods, such as RNN [23] and CNN. Table 5 shows the comparison results of the proposed SAAE-DNN and other models in the accuracy of binary-classification and multi-classification, all of which are verified on KDDTest+ and KDDTest-21 test sets. On the KDDTest + test set, the results of binary-classifications and multi-classifications are 87.74% and 82.97%, respectively. On the KDDTest-21 test set, the results of binary-classifications and multi-classifications

are 82.14% and 77.57%, respectively. Compared with other methods, this method achieves the best effect in the accuracy of binary-classification and multi-classification and has better network intrusion detection performance.

Table 5. Accuracy(%) based on NSL-KDD KDDTest+ and NSL-KDD KDDTest-21 (N/A means no available results).

Methods	Binary-Classification		Multi-Classification	
	KDDTest+	KDDTest-21	KDDTest+	KDDTest-21
RNN [23]	83.28	68.55	81.29	64.67
CNN [40]	N/A	N/A	79.48	60.71
ResNet [41]	79.14	81.57	N/A	N/A
GoogLeNet [41]	77.04	81.84	N/A	N/A
MDPCA-DBN [42]	N/A	N/A	82.08	66.18
SAE-SVM [43]	84.96	N/A	80.48	N/A
SAAE-DNN	87.74	82.97	82.14	77.57

6. Conclusions and Future Work

In this paper, we propose an intrusion detection method, SAAE-DNN, which combines SAE, attention mechanism, and DNN. For high-dimensional data, SAAE can mine potential key features of data. At the same time, the trained latent layer of SAAE is used to initialize the weights of the DNN hidden layer. The intrusion detection performance of SAAE-DNN is evaluated on the NSL-KDD dataset and compared with six famous machine learning algorithms. Experimental results show that, compared with six commonly used machine learning algorithms such as DT, XGBoost, LightGBM, GBDT, LR, and RF, the SAAE-DNN proposed in this paper has a better intrusion detection effect. Compared with existing classifiers (such as RNN, CNN, ResNet, GoogLeNet, MDPCA-DBN, and SAE-SVM), SAAE-DNN also has higher accuracy. Experiments show that SAAE-DNN is more suitable for network intrusion detection.

In future work, we plan to study a more effective method to improve the performance of IDS. We plan to study more types of AE and their latent layer to better extract key features, such as variational autoencoder. For some attacks with few samples, adversarial learning method is considered, which can synthesize similar attacks and increase the diversity of training samples to improve the accuracy of detecting them. In addition, in order to improve the training and detection efficiency of the model, we will explore the use of tools such as Spark.

Author Contributions: conceptualization, C.T. and N.L.; methodology, C.T. and N.L.; software, Y.Z.; validation, C.T., Y.Z. and N.L.; formal analysis, C.T.; investigation, C.T. and Y.Z.; resources, C.T. and N.L.; data curation, Y.Z.; writing—original draft preparation, C.T.; writing—review and editing, C.T. and N.L.; visualization, Y.Z.; supervision, N.L.; project administration, N.L.; funding acquisition, N.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61433012, and in part by the Innovation Environment Construction Special Project of Xinjiang Uygur Autonomous Region under Grant PT1811.

Acknowledgments: First of all, we are very grateful to all the reviewers who helped review the manuscript and put forward very valuable suggestions to improve the quality of the manuscript. We also thank the staff of the Symmetry Editorial Office for completely maintaining strict peer review work arrangements and publishing them in a timely manner.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IDS	Intrusion Detection System
AE	Autoencoder
SAE	Staked Autoencoder
DAE	Denoising Autoencoder
R2L	Remote-to-Local
U2R	User-to-Root
DoS	Denial-of-Service
AAE	Attention-AutoEncoder
SAAE	Attention-Stacked-AutoEncoder
DNN	Deep Neural Networks
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives

References

1. Liu, C.; Liu, Y.; Yan, Y.; Wang, J. An Intrusion Detection Model With Hierarchical Attention Mechanism. *IEEE Access* **2020**, *8*, 67542–67554. [CrossRef]
2. Dwivedi, S.; Vardhan, M.; Tripathi, S.; Shukla, A.K. Implementation of adaptive scheme in evolutionary technique for anomaly-based intrusion detection. *Evol. Intell.* **2020**, *13*, 103–117. [CrossRef]
3. Su, T.; Sun, H.; Zhu, J.; Wang, S.; Li, Y. BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset. *IEEE Access* **2020**, *8*, 29575–29585. [CrossRef]
4. Alagrash, Y.; Drebee, A.; Zirjawi, N. Comparing the Area of Data Mining Algorithms in Network Intrusion Detection. *J. Inf. Secur.* **2020**, *11*, 1–18. [CrossRef]
5. Khammassi, C.; Krichen, S. A NSGA2-LR wrapper approach for feature selection in network intrusion detection. *Comput. Netw.* **2020**, *172*, 107183. [CrossRef]
6. Gauthama Raman, M.R.; Somu, N.; Jagarapu, S.; Manghnani, T.; Selvam, T.; Krithivasan, K.; Shankar Sriram, V.S. An efficient intrusion detection technique based on support vector machine and improved binary gravitational search algorithm. *Artif. Intell. Rev.* **2020**, *53*, 3255-–3286. [CrossRef]
7. Ieracitano, C.; Adeel, A.; Morabito, F.C.; Hussain, A. A Novel Statistical Analysis and Autoencoder Driven Intelligent Intrusion Detection Approach. *Neurocomputing* **2020**, *387*, 51–62. [CrossRef]
8. Dey, S.K.; Rahman, M.M. Effects of Machine Learning Approach in Flow-Based Anomaly Detection on Software-Defined Networking. *Symmetry* **2020**, *12*, 7. [CrossRef]
9. Elmasry, W.; Akbulut, A.; Zaim, A.H. Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. *Comput. Netw.* **2020**, *168*, 107042. [CrossRef]
10. Iwendi, C.; Khan, S.; Anajemba, J.H.; Mittal, M.; Alenezi, M.; Alazab, M. The Use of Ensemble Models for Multiple Class and Binary Class Classification for Improving Intrusion Detection Systems. *Sensors* **2020**, *20*, 2559. [CrossRef] [PubMed]
11. Mikhail, J.W. ; Fossaceca, J.M.; Iammartino, R. A semi-boosted nested model with sensitivity-based weighted binarization for multi-domain network intrusion detection. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–27. [CrossRef]
12. Kumar, G. An improved ensemble approach for effective intrusion detection. *J. Supercomput.* **2020**, *76*, 275–291. [CrossRef]
13. Safara, F.; Souri, A.; Serrizadeh, M. Improved intrusion detection method for communication networks using association rule mining and artificial neural networks. *IET Commun.* **2020**, *14*, 1192–1197. [CrossRef]
14. Amouri, A.; Alaparthy, V.T.; Morgera, S.D. A Machine Learning Based Intrusion Detection System for Mobile Internet of Things. *Sensors* **2020**, *20*, 461. [CrossRef] [PubMed]
15. Hu, B.; Wang, J.; Zhu, Y.; Yang, T. Dynamic Deep Forest: An Ensemble Classification Method for Network Intrusion Detection. *Electronics* **2019**, *8*, 968. [CrossRef]

16. Velliangiri, S. A hybrid BGWO with KPCA for intrusion detection. *J. Exp. Theor. Artif. Intell.* **2020**, *32*, 165–180. [CrossRef]
17. Karthikeyan, D.; Mohanraj, V.; Suresh, Y.; Senthilkumar, J. Hybrid Intrusion Detection System Security Enrichment Using Classifier Ensemble. *J. Comput. Theor. Nanosci.* **2020**, *17*, 434–438. [CrossRef]
18. Wongsuphasawat, K.; Smilkov, D.; Wexler, J.; Wilson, J.; Mané, D.; Fritz, D.; Krishnan, D.; Viégas, F.B.; Wattenberg, M. Visualizing dataflow graphs of deep learning models in TensorFlow. *IEEE Trans. Vis. Comput. Graph.* **2018**, *24*, 1–12. [CrossRef]
19. He, D.; Qiao, Q.; Gao, Y.; Zheng, J.; Chan, S.; Li, J.; Guizani, N. Intrusion Detection Based on Stacked Autoencoder for Connected Healthcare Systems. *IEEE Netw.* **2019**, *33*, 64–69. [CrossRef]
20. Du, S.; Li, T.; Yang, Y.; Horng, S. Multivariate Time Series Forecasting via Attention-based Encoder-Decoder Framework. *Neurocomputing* **2020**, *388*, 269–279. [CrossRef]
21. Chiba, Z.; Abghour, N.; Moussaid, K.; Omri, A.E.; Rida, M. Intelligent approach to build a Deep Neural Network based IDS for cloud environment using combination of machine learning algorithms. *Comput. Secur.* **2019**, *86*, 291–317. [CrossRef]
22. Javaid, A.Y.; Niyaz, Q.; Sun, W.; Alam, M. A Deep Learning Approach for Network Intrusion Detection System. *EAI Endorsed Trans. Secur. Saf.* **2016**, *3*, 21–26.
23. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
24. Anwer, H.M.; Farouk, M.; Abdel-Hamid, A. A framework for efficient network anomaly intrusion detection with features selection. In Proceedings of the 2018 9th International Conference on Information and Communication Systems, Irbid, Jordan, 3–5 April 2018; pp. 157–162.
25. Mighan, S.N.; Kahani, M. A novel scalable intrusion detection system based on deep learning. *Int. J. Inf. Secur.* **2020**, 1–17. [CrossRef]
26. Jo, W.; Kim, S.; Lee, C.; Shon, T. Packet Preprocessing in CNN-Based Network Intrusion Detection System. *Electronics* **2020**, *9*, 1151. [CrossRef]
27. Mnih, V.; Heess, N.; Graves, A.; Kavukcuoglu, K. Recurrent Models of Visual Attention. In Proceedings of the Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2204–2212.
28. Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; Torralba, A. Learning Deep Features for Discriminative Localization. In Proceedings of the Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2921–2929.
29. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In Proceedings of the International Conference on Machine Learning, Lille, French, 6–11 July 2015; pp. 2048–2057.
30. Yang, T.; Hu, Y.; Li, Y.; Hu, W.; Pan, Q. A Standardized ICS Network Data Processing Flow With Generative Model in Anomaly Detection. *IEEE Access* **2020**, *8*, 4255–4264. [CrossRef]
31. Kunang, Y.N.; Nurmaini, S.; Stiawan, D.; Zarkasi, A.; Jasmir, F. Automatic Features Extraction Using Autoencoder in Intrusion Detection System. In Proceedings of the International Conference on Electrical Engineering and Computer Science (ICECOS), Pangkal Pinang, Indonesia, 2–4 October 2018; pp. 219–224.
32. Montañez, C.A.C.; Fergus, P.; Chalmers, C.; Malim, N.H.A.H.; Abdulaimma, B.; Reilly, D.; Falciani, F. SAERMA: Stacked Autoencoder Rule Mining Algorithm for the Interpretation of Epistatic Interactions in GWAS for Extreme Obesity. *IEEE Access* **2020**, *8*, 112379–112392. [CrossRef]
33. Swetha, A. Churn Prediction using Attention Based Autoencoder Network. *Int. J. Adv. Trends Comput. Sci. Eng.* **2019**, *8*, 725–730.
34. Feng, F.; Liu, X.; Yong, B.; Zhou, R.; Zhou, Q. Anomaly detection in ad-hoc networks based on deep learning model: A plug and play device. *Ad Hoc Netw.* **2019**, *84*, 82–89. [CrossRef]
35. Khare, N.; Devan, P.; Chowdhary, C.L.; Bhattacharya, S.; Singh, G.; Singh, S.; Yoon, B. SMO-DNN: Spider Monkey Optimization and Deep Neural Network Hybrid Classifier Model for Intrusion Detection. *Electronics* **2020**, *9*, 692. [CrossRef]
36. Duan, B.; Han, L.; Gou, Z.; Yang, Y.; Chen, S. Clustering Mixed Data Based on Density Peaks and Stacked Denoising Autoencoders. *Symmetry* **2019**, *11*, 163. [CrossRef]
37. Tharwat, A. Classification assessment methods. *Appl. Comput. Inform.* **2018**, *10*, 1–13. [CrossRef]

38. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
39. Yang, Y.; Zheng, K.; Wu, B.; Yang, Y.; Wang, X. Network Intrusion Detection Based on Supervised Adversarial Variational Auto-Encoder With Regularization. *IEEE Access* **2020**, *8*, 42169–42184. [CrossRef]
40. Wu, K.; Chen, Z.; Li, W. A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks. *IEEE Access* **2018**, *6*, 50850–50859. [CrossRef]
41. Li, Z.; Qin, Z.; Huang, K.; Yang, X.; Ye, S. Intrusion Detection Using Convolutional Neural Networks for Representation Learning. In Proceedings of the International Conference on Neural Information Processing, Guangzhou, China, 14–18 October 2017; pp. 858–866
42. Yang, Y.; Zheng, K.; Wu, C.; Niu, X.; Yang, Y. Building an Effective Intrusion Detection System Using the Modified Density Peak Clustering Algorithm and Deep Belief Networks. *Appl. Sci.* **2019**, *9*, 238. [CrossRef]
43. Alqatf, M.; Lasheng, Y.; Alhabib, M.; Alsabahi, K. Deep Learning Approach Combining Sparse Autoencoder With SVM for Network Intrusion Detection. *IEEE Access* **2018**, *6*, 52843–52856. [CrossRef]